End-User Programming of Manipulator Robots in Situated Tangible Programming Paradigm

Yasaman S. Sefidgar
University of Washington
einsian@gmail.com

Maya Cakmak
University of Washington
mcakmak@cs.washington.edu

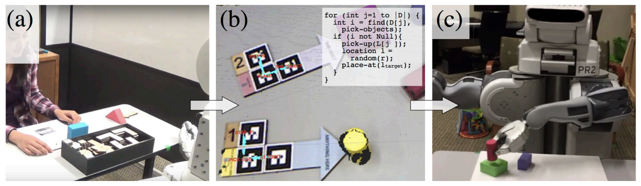
ABSTRACT

While the cost of creating robots is declining, deploying them in industry remains expensive. Widespread use of robots, particularly in smaller industries, is more easily realized if robot programming is accessible to non-programmers. Our research explores techniques to lower the barrier to robot programming. In one such attempt, we propose *situated tangible robot programming* to program a robot by placing specially designed tangible *blocks* in its workspace. These blocks are used for annotating objects, locations, or regions, and specifying actions and their ordering. The robot compiles a program by detecting blocks and objects in the environment and grouping them into instructions by solving constraints. We designed a preliminary tangible language and blocks and evaluated the intuitiveness and learnability of the approach. Our user studies provide evidence for the promise of situated tangible programming and identify the challenges to address. In addition to improving the block design and extending the language, we are planning to integrate tangible programming into a holistic ecosystem of a programming environment in future.

KEYWORDS

End-User Programming, Robot Programming Tools, Tangible Programming, Situated Programming

ACM Reference Format:
Yasaman S. Sefidgar and Maya Cakmak. 2018. End-User Programming of Manipulator Robots in Situated Tangible Programming Paradigm. In *HRI '18 Companion: 2018 ACM/IEEE International Conference on Human-Robot Interaction Companion, March 5–8, 2018, Chicago, IL, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3173386.3176923



```
for (int j=1 to |D|) {
  int i = find(D[j],
    pick-objects);
  if (i not Null){
    pick-up(L[j ]);
    location l =
      random(r);
    place-at(l_target);
  }
}
```

Figure 1: *(a) Situated tangible programming* involves programming a robot by combining and placing specially designed tangible blocks in the robot's workspace to select objects, locations, or regions, and to specify actions (e.g. pick or place) and their ordering. (b) Blocks and objects in the workspace are detected by the robot and compiled into a robot program. (c) The robot can perform the instructed task in new environments by executing this program even after the blocks are removed.

1 INTRODUCTION

Traditional automation is too costly and inflexible to satisfy the low volume and high mix production requirements that are commonly encountered in smaller industries. Becoming increasingly cheaper and safer, robotic manipulators hold the promise for revolutionizing industrial automation. Lower cost and programmability of robots allow industries of all sizes to benefit from automation and respond to dynamic customer needs. There is, however, a substantial barrier to realizing such promise: the languages and interfaces for programming robot manipulators are notoriously complex. Programming robots thus remains expensive, as it requires advanced knowledge and expertise and takes considerable time even for trained experts. Having identified this significant need, robotics companies now emphasize end-user programmability to reduce the cost of programming and eliminate down-time. For example, Rethink Robotics, a pioneer in this area, advertises its robots, Baxter and Sawyer, as "simple to train, flexible, and re-deployable."

Programming by Demonstration (PbD) is a popular approach to make robot programming accessible to people with little programming background. Despite considerable research in this area, there are still difficulties to address. Referencing objects or arbitrary parts of the environment relevant to the robot's actions poses a particularly difficult end-user programming challenge. Some programming systems incorporate a separate procedure to specify objects or locations that are relevant for the task and typically require the definition of coordinate frames in relation to the environment. Others have proposed "situated" approaches, such as using pointing gestures [1], verbal descriptions [5], or visual annotations [2], but lack the robustness needed in industrial settings.

In our research we explore a new way of programming robots that takes advantage of being situated in the task context and is expressive enough for the requirements of a wide range of industrial tasks. Our approach involves placing physical, tangible *blocks* in the robot's task environment to annotate objects, locations, or regions and to instruct the robot to perform actions that reference those places.

2 RESEARCH PLAN

Our plan to explore situated tangible robot programming for industrial manipulation span four areas:

Industrial Manipulation Task Analysis. We will perform field observations of industrial tasks that robots need to be programmed for. This will allow us to understand the kind of information tangible programs should express as well as the context and constraints of programming in industrial settings.

Iterative Language and Block Design. We will design the tangible programming language and its associated blocks based on the requirements of the tasks and the programming setting. Following a user-centered approach, we will iteratively evaluate intuitiveness and learnability of the language and make adjustments accordingly.

Compiler Development. We will implement an end-to-end system that detects tangible blocks, groups them into instructions, and translates those instructions to robot motions.

Programming Environment Development. We will integrate the situated tangible programming language into a holistic programming environment with verification, testing, and debugging capabilities. We explore varied interface modalities and interaction techniques that allow a seamless coordination between tangible program expression and other engineering activities.

3 PRELIMINARY FINDINGS

We have made progress in the first three areas introduced in our plan. We analyzed pick and place tasks common in industrial settings (e.g., machine tending, assembly, and packaging). Subsequently, we developed a situated tangible programming language for simple pick and place actions, designed tangible blocks that allow programming in this language, and implemented a proof-of-concept perception and execution system that compiles tangible instructions to robot motions [3, 4].

We conducted three user studies to examine the intuitiveness and learnability of the language and to identify areas for improvement. More specifically, we examined whether people can understand tangible instructions or create programs with them. We compared program comprehension and program creation performance before and after training in our first study to assess both intuitiveness and learnability. We updated our block design based on what we learned and examined the same capabilities without any training in two follow-up studies.

Our studies demonstrate that situated tangible programming allows participants to program a robot with minimal or no instruction. We cannot draw a direct comparison with instructions given to participants for other end-user robot programming techniques in the literature; however, participants' ability to program the robot without *any* instructions is unique. Further comparative studies will highlight the advantages and limitations of this approach against others and will enable hybrid interfaces that incorporate the advantages of both without suffering from the limitations.

4 FUTURE WORK

Encouraged by our preliminary efficacy evaluation, we will continue our research following our original plan (Section 2). We will first aim to obtain more in-depth understanding of tasks robots should support to refine and extend the language we have already developed. We will then expand our proof-of-concept end-to-end system to support the improved language. Performing comparative studies between tangible programming and other approaches such as text-based or visual programming (both situated and not) will be the next important step and will help define the boundaries for tangible programming and advance its integration into a comprehensive programming environment.

REFERENCES

[1] Rui Fang, Malcolm Doering, and Joyce Y Chai. 2015. Embodied collaborative referring expression generation in situated human-robot interaction. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction.* ACM, 271–278.

[2] Richard Fung, Sunao Hashimoto, Masahiko Inami, and Takeo Igarashi. 2011. An augmented reality system for teaching sequential tasks to a household robot. In *2011 RO-MAN.* IEEE, 282–287.

[3] Yasaman Sefidgar, Prerna Agarwal, and Maya Cakmak. 2017. Situated Tangible Robot Programming. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI).*

[4] Yasaman Sefidgar, Sarah Elliott, and Maya Cakmak. 2017. A System for Situated Tangible Programming of Robot Skills. In *IROS Workshop on Learning for Collaborative Robotics: Enabling Flexible, Redeployable and Agile Industrial Applications.*

[5] Lanbo She, Yu Cheng, Joyce Y Chai, Yunyi Jia, Shaohua Yang, and Ning Xi. 2014. Teaching robots new actions through natural language instructions. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication.* IEEE, 868–873.